LAB 07

CONDITIONAL PROCESSING

Syed Muhammad Faheem	20K-1054	3E
STUDENT NAME	ROLL NO	SEC

_____

SIGNATURE & DATE

MARKS AWARDED: _____

EL 2003 COMP ORG & ASSEMBLY LANGUAGE LAB

LABORATORY MANUAL • Fall 2021 • Instructor: Aamir Ali , Aashir Mehboob, Amin Sadiq, Qurat -ul-Ain, M Kariz, Rabia

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(NUCES), KARACHI

Prepared by: Aashir Mahboob

Version: 1.0

Date: 27th Oct 2021

Lab Session 07: CONDITIONAL PROCESSING

Objectives:

- Boolean Instructions
- Set Operations
- CMP Instruction
- Conditional Jumps

Boolean Instructions

• AND

Boolean AND operation between a source operand and destination operand.

Syntax:

```
AND reg, reg
AND reg, mem
AND reg, imm
AND mem, reg
AND mem, imm
```

• OR

Boolean OR operation between a source operand and destination operand.

Syntax:

```
OR reg, reg
OR reg, mem
OR reg, imm
OR mem, reg
OR mem, imm
```

• XOR

Boolean XOR operation between a source operand and destination operand.

Syntax:
XOR reg, reg
XOR reg, mem
XOR reg, imm
XOR mem, reg
XOR mem, imm

- **NOT**

Boolean NOT operation on a destination operand.

Syntax:
NOT reg
NOT mem

- **TEST**

Similar to AND operation, except that instead of affecting any operands it sets the FLAGS appropriately.

Syntax:
TEST reg, reg
TEST reg, mem
TEST reg, imm
TEST mem, reg
TEST mem, imm

Example 01:

```
Include Irvine32.inc
.code  main
proc
      mov     al, 10101110b          ; Clear only bit 3
      and     al, 11110110b          ; AL = 10100110

      mov     al,  11100011b         ; set bit 2
```

```
        or      al, 00000100b          ; AL = 11100111

        mov     al,  10110101b         ;  5 bits means odd parity
        xor     al, 0                  ; PF = 0 (PO)

        mov     al,  10100101b         ;  4 bits means even parity
        xor     al, 0                   ; PF = 1 (PE)

        mov     al,  11110000b
        not     al                              ; AL = 00001111b

        mov     al, 00100101b
        test    al, 00001001b          ; ZF = 0

        mov     al, 00100101b
        test    al, 00001000b          ; ZF = 1
        call    DumpRegs
    exit
main ENDP
END main
```

Set Operations (using Boolean instructions)

- **Set Complement**

The complement of a set can be achieved through NOT instruction.

- **Set Intersection**

The intersection of two sets can be achieved through AND instruction.

- **Set Union**

The union of two sets can be achieved through OR instruction.

Example 02:

```
Include Irvine32.inc
 .data
      A DWORD 10000000000000000000000000000111b
      B DWORD 10000001010100000000011101100011b
```

```
        msg1 BYTE "A intersection B is: ", 0          msg2
    BYTE "A union B is: ", 0
        msg3 BYTE "Complement of A is: ", 0
    .code main
    proc
        mov     eax,A
        and     eax, B          ;  A intersection B
        mov     edx, OFFSET msg1
        call    WriteString
        mov     ebx, TYPE DWORD
        call    WriteBinB
        call    Crlf
        mov     eax, A
        or      eax, B          ;  A union B
        mov     edx, OFFSET msg2
        call    WriteString
        mov     ebx, TYPE DWORD
        call    WriteBinB
        call    Crlf
        mov     eax, A
        not     eax                     ; A complement
        mov     edx, OFFSET msg3
        call    WriteString
        mov     ebx, TYPE DWORD
        call    WriteBinB
    call    DumpRegs
    exit
    main ENDP END
    main
```

CMP instruction

CMP (compare) instruction performs an implied subtraction of a source operand from a

destination operand for comparison.
For unsigned operands:

- Destination < source ZF = 0 CF = 1
- Destination > source ZF = 0 CF = 0

- Destination = source ZF = 1 CF = 0

For signed operands:

- Destination < source SF ! = OF
- Destination > source SF = OF
- Destination = source ZF = 1

Example 03:

```
Include Irvine32.inc
.code
main proc
        mov    ax, 5
        cmp    ax, 10          ;  ZF = 0      and     CF = 1
        mov    ax, 1000
        cmp    ax,  1000       ; ZF = 1       and     CF =0
        mov    si, 106
        cmp    si, 0           ; ZF = 0       and     CF = 0
call    DumpRegs
 exit
main ENDP
END main
```

Conditional Jumps

- **Jumps based on Flag values**

Mnemonic	Description	Flags / Registers
JZ	Jump if zero	ZF = 1
JNZ	Jump if not zero	ZF = 0
JC	Jump if carry	CF = 1
JNC	Jump if not carry	CF = 0
JO	Jump if overflow	OF = 1
JNO	Jump if not overflow	OF = 0
JS	Jump if signed	SF = 1
JNS	Jump if not signed	SF = 0
JP	Jump if parity (even)	PF = 1
JNP	Jump if not parity (odd)	PF = 0

- **Jumps based on Equality**

Mnemonic	Description
JE	Jump if equal (*leftOp* = *rightOp*)
JNE	Jump if not equal (*leftOp* ≠ *rightOp*)
JCXZ	Jump if CX = 0
JECXZ	Jump if ECX = 0

- **Jumps based on unsigned comparisons**

Mnemonic	Description
JA	Jump if above (if *leftOp* > *rightOp*)
JNBE	Jump if not below or equal (same as JA)
JAE	Jump if above or equal (if *leftOp* ≥ *rightOp*)
JNB	Jump if not below (same as JAE)
JB	Jump if below (if *leftOp* < *rightOp*)
JNAE	Jump if not above or equal (same as JB)
JBE	Jump if below or equal (if *leftOp* ≤ *rightOp*)
JNA	Jump if not above (same as JBE)

- **Jumps based on signed comparisons**

Mnemonic	Description
JG	Jump if greater (if *leftOp* > *rightOp*)
JNLE	Jump if not less than or equal (same as JG)
JGE	Jump if greater than or equal (if *leftOp* ≥ *rightOp*)
JNL	Jump if not less (same as JGE)
JL	Jump if less (if *leftOp* < *rightOp*)
JNGE	Jump if not greater than or equal (same as JL)
JLE	Jump if less than or equal (if *leftOp* ≤ *rightOp*)
JNG	Jump if not greater (same as JLE)

Example 04:

```
Include Irvine32.inc
 .data
      var1 DWORD 250
      var2 DWORD 125
      larger DWORD ?
.code  main
proc
      mov    eax, var1
      mov    larger, eax
      mov    ebx, var2
```

```
        cmp     eax, ebx
        jae     L1
        mov     larger, ebx
L1: call    DumpRegs exit
main ENDP
END main
```

Example 05:

```
Include Irvine32.inc
 .data
        var1    DWORD 50
        var2    DWORD 25
        var3    DWORD 103
        msg     BYTE "The smallest integer is: ", 0
.code  main
proc mov
eax, var1
 cmp eax, var2  jbe L1
        mov     eax, var2
        L1:

        cmp     eax, var3
jbe L2  mov eax, var3

        L2:
        mov     edx, OFFSET msg
```

```
        call    WriteString
        call    WriteDec
    call    DumpRegs
    exit
    main ENDP
    END main
```

Example 06:

```
    Include Irvine32.inc
     .data
    char BYTE ?
    .code  main
    proc L1:


     mov  eax,  10      call            ; create 10ms delay
    Delay
        call    ReadKey                 ; reads a key input
        jz      L1                      ; repeat if no key is pressed
     mov  char,  al  call          ; saves the character
    DumpRegs  exit
    main ENDP
    END main
```

Lab Task(s):

1. Translate the following pseudo-code to Assembly Language:

```
        var = 5
        if ( var<ecx ) AND          (ecx>=edx)
then            x = 0   else        x = 1
```

2. Use cmp and jumps to find the first non-zero value in the given array:

   intArr SWORD 0, 0, 0, 0, 1, 20, 35, -12, 66, 4, 0

3. Write a program that takes four input integers from the user. Then compare and display a message whether these integers are equal or not.

4. Write a program for sequential search. Take an input from the user and find if it occurs in the following array:

   arr WORD 10, 4, 7, 14, 299, 156, 3, 19, 29, 300, 20

5. Translatethe followingpseudo-codeto Assembly Language:

```
            Swap_Count = 0
            for all elements of list
                if list[i] > list[i+1]
swap(list[i], list[i+1])                    Swap_Count =
Swap_Count + 1
                    end if
        end for         Print
Swap_Count
```

Task 1:

```
include irvine32.inc
.model small
.stack 100h
.data
var dword 5
x byte 5
.code
main proc
;mov ecx,2
;mov edx,3
cmp var,ecx
JB L1
L1:
cmp ecx,edx
JAE L2
mov x,0
mov eax,0
mov al,x
call WriteDec
invoke exitprocess,0
L2:
mov x,1
mov eax,0
mov al,x
call WriteDec
exit
main endp
end main
```

```
Microsoft Visual Studio Debug Console
1
C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 25608) exited with code 0.
Press any key to close this window . . .
```

Task 2:

```
include irvine32.inc
.model small
.stack 100h
.data
intArr sword 0,0,0,0,1,20,35,-12,66,4,0
msg1 byte "The first non-zero value is: ",0
msg2 byte "The non-zero value is at index: ",0
index_count byte -1
.code
main proc
mov esi,0
L1:
movzx eax,intArr[esi]
add esi,2
inc index_count
cmp eax,0
JE L1
mov edx,offset msg1
call WriteString
call WriteInt
call Crlf
mov edx,offset msg2
call WriteString
movzx eax,index_count
call WriteInt
exit
main endp
end main
```

```
Microsoft Visual Studio Debug Console
The first non-zero value is: +1
The non-zero value is at index: +4
C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 2520
Press any key to close this window . . .
```

Task 3:

```
include irvine32.inc
.model small
.stack 100h
.data
var1 dword ?
var2 dword ?
var3 dword ?
var4 dword ?
msg1 byte "Enter the integer: ",0
msg2 byte "The inserted integers entered are equal! ",0
msg3 byte "The inserted integers entered are not equal! ",0
.code
main proc
mov edx,offset msg1
call WriteString
call ReadInt
mov var1,eax
mov edx,offset msg1
call WriteString
call ReadInt
mov var2,eax
mov edx,offset msg1
call WriteString
call ReadInt
mov var3,eax
mov edx,offset msg1
call WriteString
call ReadInt
mov var4,eax
mov ebx,var1
mov ebx,var2
mov ebx,var3
mov ebx,var4
mov ecx,4
L1:
sub ebx,var1
L2:
cmp ebx,0
JE L3
mov edx,offset msg3
call WriteString
exit
L3:
mov edx,offset msg2
call WriteString
exit
main endp
```

```
Microsoft Visual Studio Debug Console
Enter the integer: 5
Enter the integer: 5
Enter the integer: 5
Enter the integer: 5
The inserted integers entered are equal!
C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 10112) exited with code 0.
Press any key to close this window . . .
```

Task 4:

```
include irvine32.inc
.model small
.stack 100h
.data
arr word 10,4,7,14,299,156,3,19,29,300,20
var word ?
msg1 byte "Enter the integer for Sequential Search: ",0
msg2 byte "The Searched integer was found!",0
msg3 byte "The Searched integer was not found!",0
.code
main proc
mov esi,offset arr
mov edx,offset msg1
call WriteString
mov eax,0
call ReadInt
mov var,ax
mov ecx,lengthof arr
L1:
mov eax,0
mov ax,[esi]
add esi,2
cmp ax,var
JE L2
loop L1
mov edx,offset msg3
call WriteString
exit
L2:
mov edx,offset msg2
call WriteString
call WriteInt
exit
main endp
end main
```

```
Microsoft Visual Studio Debug Console
Enter the integer for Sequential Search: 19
The Searched integer was found!+19
C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 22620) exited with code 0.
Press any key to close this window . . .
```

Task 5:

```
include irvine32.inc
.model small
.stack 100h
.data
arr dword 5,4,3,2,1
swap_count dword 0
msg1 byte "The Swap Count of the whole array is: ",0
.code
main proc
mov esi,0
mov edi,4
mov ecx,(lengthof arr-1)
L1:
mov eax,arr[esi]
mov ebx,arr[edi]
cmp eax,ebx
JA L2
add esi,4
add edi,4
loop L1
L2:
inc swap_count
mov arr[esi],ebx
mov arr[edi],eax
dec ecx
cmp ecx,0
JECXZ L3
add esi,4
add edi,4
JMP L1
L3:
mov edx,offset msg1
call WriteString
mov eax,swap_count
call WriteInt
call Crlf
mov ecx,lengthof arr
mov esi,0
L4:
mov eax,arr[esi]
call WriteInt
add esi,4
loop L4
exit
main endp
end main
```

```
Microsoft Visual Studio Debug Console
The Swap Count of the whole array is: +4
+4+3+2+1+5
C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 29632) exited with code 0.
Press any key to close this window . . .
```